# Mono-Static RCS Computation with a Block GMRES Iterative Solver

WILLIAM E. BOYSE
Advanced Software Resources
3375 Scott Boulevard, Suite 420, Santa Clara, CA 95054
boyse@netcom.com

ANDREW A. SEIDL
Lockheed-Martin Missles and Space Co.
3251 Hanover Street, Palo Alto, CA 94034
seidl@firenze.lmsc.lockheed.com

## Abstract

We present a new method of computing mono-static radar cross sections using a preconditioned Block GMRES iterative algorithm. The convergence properties of this algorithm are analyzed using RCS error, equation residual error and solution error. It is found that this method is nearly an order of magnitude faster than direct methods (LINPACK) for realistic method of moment problems.

## 1 Introduction

Iterative methods have long been the solver of choice for sparse matrix problems coming primarily from finite element or finite difference techniques. In electromagnetic scattering applications, these solvers quickly provide bistatic RCS results, but are time consuming for full mono-static data generation. Two approaches have been used to make iterative solvers more efficient for multiple right-hand sides and thus mono-static RCS calculations. They both involve using information generated from the iterates of one right-hand side to assist in solving others.

In [1, 2] a "seed" system is iterated and other right-hand side vectors are projected onto the resulting Krylov sub-space to approximate those solutions. Analyzed in [3], the method in [1] converges best when the right-hand side vectors are close together. For mono-static RCS calculations, this refers to right-hand sides representing incidence angles which are close. This limits the usefulness of this method since typically mono-static RCS is computed over a broad range of angles.

The other approach [4, 5, 6, 7, 8], is to use block iterative methods, i.e., directly solve multiple right-hand sides simultaneously. These methods use the iterates from all the right-hand sides in the block to approximate all their solutions, and as a result converge significantly faster than methods using only a single right-hand side.

When applying such iterative methods to dense matrices as encountered with the method of moments (MOM), the first obstacle is the loss of symmetry (either real or complex). While there are generalizations of usual iterative methods for non-Hermitian matrices, the benchmark algorithm for such matrices is GMRES [9]. It is appropriate for general matrices and directly minimizes the residual error which provides a monotonically decreasing convergence criterion.

The disadvantage of the GMRES algorithm is that one must store all the Krylov vectors, rather than only the current and previous two as in the familiar conjugate gradient algorithm. Therefore, the storage goes up linearly with the iterations, and can become prohibitive for large problems. The algorithm can be restarted, using the current iterate as the initial guess and starting the algorithm anew, but this slows convergence considerably and the solution is no longer the minimum residual over the whole Krylov sub-space.

The approach taken in this paper is to obtain convergence as rapidly as possible, thus minimizing the storage needed. This is achieved by using a sparse incomplete factorization preconditioner and by using the block algorithm. Then we make maximum use of the orthonormal basis of the Krylov sub-space which is computed and stored when running the GMRES algorithm. We note that others have taken a similar approach [10, 3].

We use this block algorithm to solve for equally spaced mono-static points over a range of angles. The interme-

diate mono-static RCS values are computed as minimum residual solutions to the equations using the previously computed Krylov sub-space vectors. By examining in detail the RCS error, the equation residual and the solution error, we show that the efficiency of this method is near the optimum, as indicated by the Nyquist sampling rate. This technique provides additional solutions without any additional matrix multiplications or preconditioner applications, making them virtually free.

This combination of techniques is shown to compute mono-static RCS, for a realistic MOM problem, almost an order of magnitude faster than single precision LU factorization by LINPACK.

## 2 Preconditioned Block GMRES Algorithm

This algorithm addresses the solution of the matrix equation $Ax = b$, where $A$ is a $n$ by $n$ complex general matrix, $x$ is an $n$ by $m$ matrix, and $b$ is an $n$ by $m$ matrix. The GMRES algorithm is described fully in numerous places, [9, 11], and will not be explicitly discussed. The only changes to the published algorithms is that in this case, the scalars become $m$ by $m$ matrices, vectors are $n$ by $m$ matrices, and the upper Hessenberg matrix in GMRES becomes block $m$ by $m$ upper Hessenberg.

The GMRES algorithm minimizes the equation residual over a sub-space of the solution space ($C^n$). At iteration $k$, the algorithm has generated an orthonormal basis for the Krylov sub-space $K_k = span\{b, Ab, \ldots, A^{k-1}b\}$, where *span* means all complex linear combinations of all columns of the $k$ $n$ by $m$ matrices in the set. Specifically, GMRES finds the iterate $x^j$ in the Krylov subspace $K_k$ that minimizes the equation residual $\|Ax^j - b^j\|$ for $j = 1, \ldots, m$, where $x^j$ refers to the "jth" column of $x$ and $\|x\|$ is Euclidean ($L_2$) norm of $x$.

Note that the Krylov sub-space $K_k$ is formed using all the columns of $b$. This permits information generated by **all** the simultaneous solutions to assist in approximating **each** solution vector. This differs for the usual, one solution at a time, algorithm where the solution vector is restricted to the Krylov sub-space generated by only its right hand side vector. This "information sharing" [2] is what accounts for the acceleration of convergence of the block algorithm.

We precondition this system with a sparse incomplete LU preconditioner. That is, we actually solve a transformed system $\tilde{A}\tilde{x} = b$, where $\tilde{A} = AU^{-1}L^{-1}$ and $\tilde{x} = LUx$, where $L$ and $U$ are the sparse incomplete lower and upper factorization matrices. Having solved for $\tilde{x}$, we may compute the original solution as $x = U^{-1}L^{-1}\tilde{x}$. For notational simplicity we will drop the tilde notation with the understanding that the preconditioning is going on in the background.

The preconditioner used is one we and others have found exceedingly effective in sparse matrix applications [8, 12]. It is easily constructed by taking a generic LU factorization algorithm and, as each row and column of $L$ and $U$ are generated, only the "large" terms are kept, thus making the rows and columns of $L$ and $U$ sparse. The criterion for "large" we found most effective is that they exceed a certain fraction of the diagonal entry. That is, matrix element $l_{ij}$ of $L$ is kept if $l_{ij} \geq fraction * l_{kk}$ where $k = \min(i, j)$ and the matrix element $u_{ij}$ of $U$ is kept if $u_{ij} \geq fraction * 1.0$ (the diagonal of $U$ is identically 1.0) where *fraction* is a non negative number typically around 0.01.

In addition, partial pivoting is performed in this calculation in an attempt to insure stability of the incomplete factorization. These types of factorizations can be unstable, resulting in no algorithmic convergence, with or without pivoting. We have not evaluated the effectiveness of pivoting in promoting stability since instances where the algorithm is unstable are rare. However, it does not cost much computationally and is the numerically prudent procedure.

## 3 Numerical Properties

In order to briefly outline the general characteristics of the preconditioned Block GMRES algorithm we present several examples. They involve the solution of a rank 1024 general complex dense matrix generated by a 3-D method of moments code. The sampling rate is nominally ten to twenty triangular patches per wavelength. It is important that the sampling rate is in the range of normal use, for we found that grossly under or oversampled problems converged much differently than normally sampled ones.

The effects of varying the number of simultaneous solutions, $m$, on the convergence rate are shown in Fig. 1. Here, a single solution is obtained, five solutions obtained simultaneously and ten solutions obtained simultaneously. There is clearly accelerated convergence with increasing block size, although not as much as experienced with sparse matrices and reported in [6, 7, 8] where the number of iterations required were reduced by a factor nearly equal to the block size.

Note that for a block size of $m$ requiring $n$ iterations, $m \times n$ matrix vector multiplications are performed, but that $m$ solutions are produced so that the number of matrix vector multiplications per solution is the same as the number of iterations. Therefore, the numerical work,

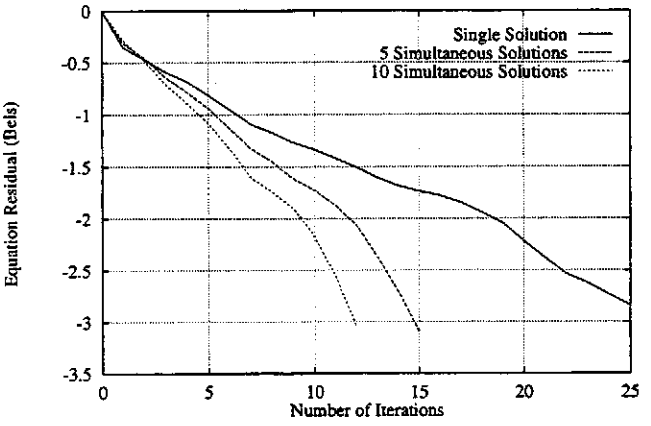


Figure 1: Convergence rates as a function of block size.

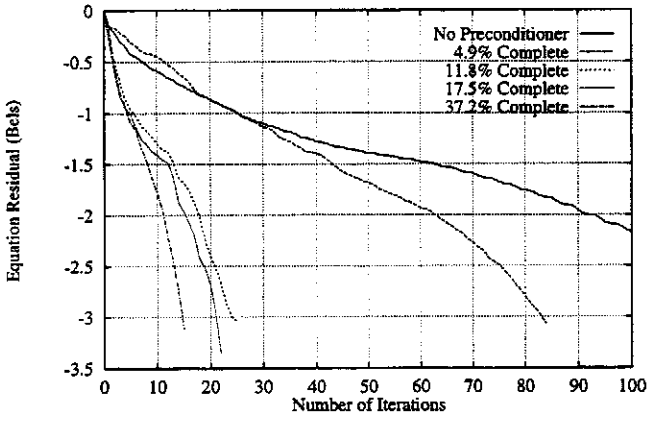


Figure 2: Convergence rates as a function of preconditioner completeness.

represented mainly by matrix vector multiplications, also decreases with increasing block size.

To demonstrate the effect of preconditioner completeness, we have adjusted the "fraction" parameter to obtain incomplete LU factorizations of varying degrees of completeness. A 10% complete preconditioner is one where the number of non-zeroes of $L$ plus those of $U$ are 10% of $n^2$ (the number in the matrix $A$). The more complete the preconditioner, the closer it is to the LU factors of the matrix and a 100% complete preconditioner is exact and will cause convergence in 0 iterations.

Figure 2 shows, to no one's surprise, that the better the preconditioner the faster the convergence. However, note that most of the benefit is achieved in the 11.8% complete case. Using a more complete preconditioner than this gives little additional reduction in the number of iterations. Considering the fact that the time spent computing the preconditioner is very close to the completeness percentage multiplied by the dense factorization time, completeness percentages around 10% appear

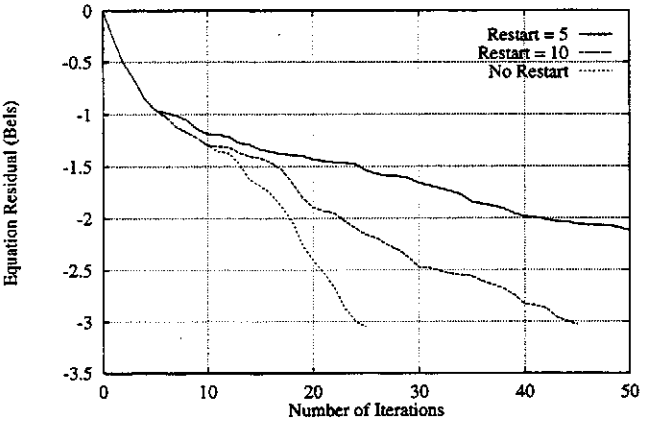


Figure 3: Convergence rates as a function of restart interval

optimal. This has held true in general and in practice we use this range of completeness.

If storage of the Krylov vectors becomes prohibitive, the algorithm can be restarted. This is accomplished simply by discarding the computed Krylov sub-space, using the current iterate as the initial guess and starting the algorithm from scratch.

To demonstrate the behavior of the algorithm when restarted, we consider solving one solution without restarting, restarting every 10 iterations and restarting every 5 iterations. ILU(T) preconditioning was used where the factors **U** and **L** are both 11.8% complete.

Figure 3 shows clearly the deleterious effects of restarting. The lower curve is the convergence without restarting. The convergence rate of both the middle and upper curves, restarting at 10 and 5 iteration intervals respectively, slows each time the algorithm is restarted. Convergence of GMRES without restarting is similar to a conjugate gradient algorithm, while with restarting every iteration reduces to simple gradient descent, a notoriously bad method. This behavior is well known and serves to emphasize that convergence must be obtained quickly so that restarting is not necessary.

# 4 Mono-static RCS Calculation

Faced with the task of solving $Ax = b$ with possibly hundreds of right-hand side vectors in a typical mono-static RCS, the block algorithm helps tremendously, but not in an obvious way. Blindly taking some or all of the right-hand side vectors and iterating runs into trouble quickly. It is no surprise that the right-hand side vectors for closely spaced angles are nearly linearly dependent. This causes both numerical problems and slow conver-

gence in the block algorithm.

Numerical difficulties stem from the orthogonalization of the Krylov vectors which, if nearly linearly dependent, is unstable (especially if a large number of simultaneous solutions are attempted). While there are schemes for "deflating" the system, i.e. dynamically removing linearly dependent vectors [10], this is a situation to be avoided.

The slow convergence is also due to this near linear dependence, causing the corresponding Krylov sub-spaces to be nearly the same, thus not "enriching" each other. This behavior was analyzed numerically and theoretically in [3] where it was shown that block iterative methods are most advantageously applied when the right-hand side vectors in the block are widely separated. Our scheme, used to calculate the mono-static RCS, avoids both numerical troubles and slow convergence.

Consider computing the mono-static RCS for angles 0° to 180° every degree. We take as a block, for example, the right-hand side vectors for 0°, 10°, ..., 180° (stride = 10°). The right-hand side vectors associated with these widely separated angles are very linearly independent. Convergence will be rapid with no numerical difficulties. The solutions corresponding to the intermediate angles will be "filled in" the following way.

The task which requires all the computational effort in the GMRES algorithm is the computation of an orthonormal basis for the Krylov sub-space for which the matrix has an upper block Hessenberg representation. We make maximum use of this wealth of information contained in this basis and representation of the matrix.

Once this basis and matrix representation are computed, the minimum residual solution to **any** right-hand side vector, not just those involved in the computation of the Krylov sub-space, may be computed easily and efficiently by projecting onto the Krylov sub-space. Therefore, the minimum residual solutions to the right-hand side vectors corresponding to intermediate angles are computed using the Krylov sub-space computed from the primary right-hand side vectors.

We apply this algorithm to the computation of the mono-static RCS, as described above, of a perfectly conducting frustrum $4\lambda$ long with end diameters of $\frac{1}{2}\lambda$ and $1\lambda$. Using a sampling rate of 10 to 20 triangles per wavelength, our 3-D MOM code required 3570 unknowns.

For convergence, the ideal criterion would be to know how many iterations are required to obtain a given accuracy in the RCS values. This, however, is impossible in general since it requires knowledge of the exact RCS with which an iterative method would not be needed. The accuracy of the RCS is controlled by the solution accuracy. This too is not computable in general, again

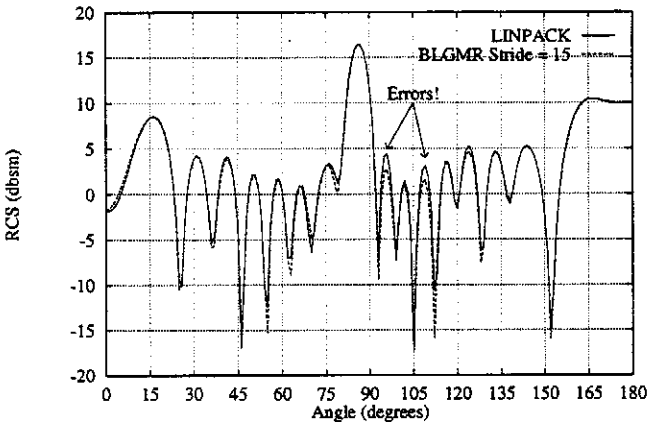


Figure 4: Mono-static RCS, Stride = 15°

because the exact solution is not known. What can be monitored is the equation residual from the algorithm, and inferences must be made of the other errors.

However, for the purposes of this evaluation, both the exact mono-static RCS and solution vectors are known (as calculated by LINPACK) and thus we may monitor the RCS and solution errors to evaluate the algorithm.

It was determined by experimentation that specifying an equation residual of $10^{-2}$ ( $-2$ Bels ) for the algorithm convergence limit was sufficient to produce accurate (overlay) RCS values and this value is used for all examples. The preconditioner completeness used was 10%, which gives the best reduction of the iteration count while minimizing the time spent computing the preconditioner. With these parameters, convergence in all examples occurred in only 10 iterations.

We will vary the interval between the angles (the stride) of the right-hand side vectors to investigate how coarsely these may be chosen and thus how efficiently the mono-static RCS may be computed. However, first we must estimate how coarse is reasonable.

Let us assume that the target is enclosed in a sphere of radius $2\lambda$ and assume that this is in the far field so that the fields are band limited to the free space wavelength on this sphere. This is not strictly true, but this estimate will bound our estimate. Sampling at the Nyquist rate of 2 samples per wavelength implies a maximum stride is somewhat less than 15°.

Figure 4 shows the comparison between the exact RCS and that generated by the iterative algorithm for a stride of 15°. The ticks on the "Angle" axis are the locations of the right-hand side vectors that are directly iterated, while the intermediate values are generated by the algorithm described above. There are unacceptable errors present in the RCS.

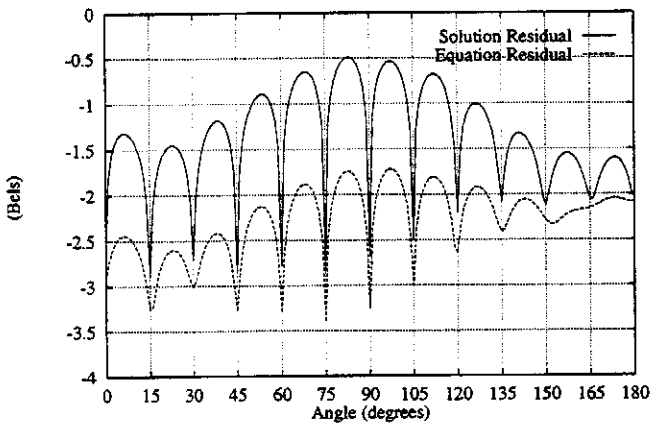


Figure 5: Equation and solution errors: stride = 15°

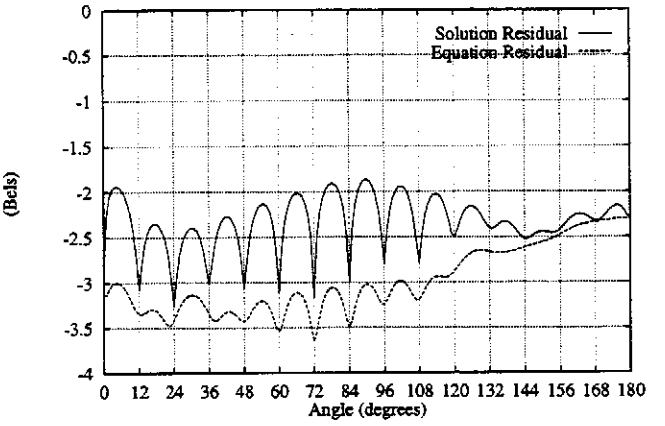


Figure 6: Equation and solution errors: stride = 12°

Figure 5 shows the relative equation residual error and the relative solution error

$$\log_{10} \frac{\|Ax_k - b\|}{\|b\|}, \quad \log_{10} \frac{\|x_k - A^{-1}b\|}{\|A^{-1}b\|} \qquad (1)$$

respectively for the stride = 15° case. Here we see that even though the equation residual (and solution error) is small at the angles directly iterated (indicated by the tick marks on the horizontal axis), it is large in between. This indicates that the stride is too large and the fields are under sampled so that the Krylov sub-space generated by the iterated angles does not span the solution vectors for the intermediate angles. This result is not totally surprising since sampling theory tells us that we have violated the Nyquist sampling theorem.

Iterating the right-hand sides in the block to an even smaller residual (−3 Bels) made no difference in these results. The residuals in between the iterated angles were still so great that accurate RCS values were not obtained.

At this point, the algorithm could be applied to the solutions that do not yet meet the error criterion (this is always an option). However, judging from the graph of residuals, virtually all the intermediate values would have to undergo further iteration and there is a more efficient way to generate these intermediate angles.

Reducing the stride to 12° produces very different results. The RCS values are in overlay agreement and are not shown. The equation residuals and solution errors are shown in Fig. 6. Here we see that reducing the residuals at the iteration angles (tick marks) also reduced the residuals of the intermediate values, enabling an accurate RCS. This value of stride is probably at or somewhat finer than the true Nyquist rate. Note that the equation residual and solution error still increase in between the iterated angles (tick marks), just not as much as in the 15° case.

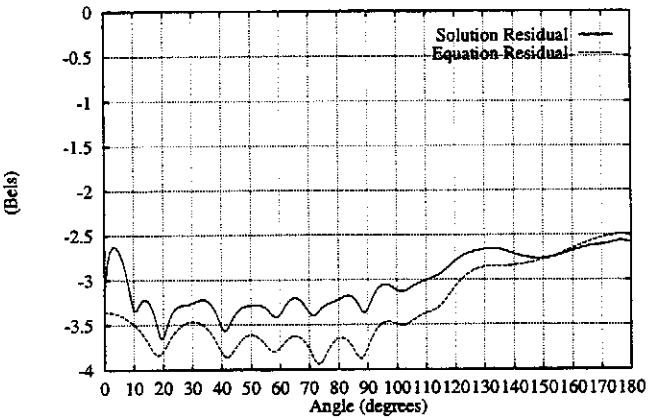


Figure 7: Equation and solution errors: stride = 10°

Finally, reducing the stride further to 10° again produces overlay RCS values. The residuals are shown in Fig. 7. Here, the pattern of the residuals does not even show characteristics of the stride, indicating that the Krylov sub-space of the iterated angles well approximates the solutions of the intermediate values.

These tests indicate that the right-hand side vectors which make up a block to be directly iterated may be sampled only slightly finer than the Nyquist rate and that the solutions for the intermediate right-hand side vectors are well approximated by the Krylov sub-space computed from the iterates.

This is an important guide in the use of this algorithm because it factors in the optical size of the scattering object. For small objects, the RCS has no character at all and no one would be surprised that it could be sampled coarsely with a large stride. However, large objects have very complicated cross sections and right-hand sides separated by too large an angle may have nothing in common at all, and may not be well approximated by each other's Krylov sub-space. This has been shown to be the

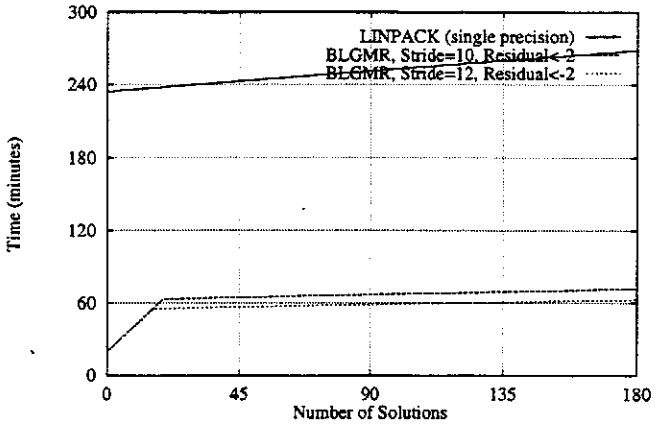


Figure 8: Wall clock time: GMRES vs Linpack

case.

This algorithm's ability to directly iterate only a subset of the desired mono-static angles has important implications for what must be the bottom line comparison, wall clock time per mono-static point.

Figure 8 shows the comparison of wall clock time versus number of mono-static points for the direct LU factorization solution method (single precision LINPACK) and the (double precision) iterative method described for both the stride = 10° and stride = 12° cases. Note that the performance of the iterative solver is not overly sensitive to number of right-hand sides. All these timings were conducted on a SUN SPARC 10 rated at approximately 17 MegaFLOPS (double precision LINPACK).

The initial time for LINPACK is how long it took to perform the LU factorization, while the slope of that curve indicates the time necessary to perform forward reduction and back substitution for each solution.

The initial time for the iterative method is the time necessary to compute the incomplete LU factorization preconditioner, a small fraction ($\sim$ 10%) of the time required for the complete LU factorization. The steep slope of the iterative curves indicates the time necessary to solve the initial block of solutions with the Block GMRES method. The final and flatter slope of these curves indicates the time required to generate the intermediate solutions using the previously computed Krylov subspace vectors. These results are generated very quickly since there are no dense matrix multiplications, preconditioner application, or Krylov vector orthogonalizations involved.

Note that final slope for the iterative solver curve is less than that of the LINPACK curve. This says that once the initial vectors are solved with the iterative method, the intermediate solutions are obtained at a faster rate than by (single precision) back substitution and forward reduction with LINPACK. Thus, if the RCS were sampled finer (unnecessary by not uncommon), the iterative solver would fare even better in comparison.

The solution time for solving each right-hand side individually with GMRES would include the preconditioner calculation time and have a slope greater than the slope for the initial block of iterated solutions. This curve would extend off the graph, being the most inefficient by far.

The iterative method (double precision) is over 4 times faster than the direct solution (single precision) on a mono-static RCS problem where it has previously been assumed that direct solution methods had the great advantage.

# 5 Summary

A preconditioned block iterative solver has been presented which computes the multiple solutions required for mono-static RCS calculations significantly faster than the universally used direct solver. The preconditioner used is of sparse incomplete type where the sparsity pattern and completeness are determined dynamically by the numerics of the factorization and not by a priori assumptions. We have found in this and other cases that this type of preconditioner is superior to those where the sparsity pattern is chosen independent of the numerics.

This iterative method is applied directly to solve right-hand side vectors corresponding to angles which differ by a constant (large) stride and subsequently the intermediate right-hand side vectors are solved using the computed Krylov sub-space. A criterion, based on Nyquist sampling theory, is given to assist in choosing the maximum stride and thus obtaining maximum efficiency.

As was shown in [3] and observed by us, the block algorithm converges faster if the right-hand side vectors in the block are not too close together. However, as was shown here, if the right-hand side vectors in the block are too far apart, the calculated Krylov sub-space does not span the solutions for the intermediate angles. This results in inaccurate approximations for the solutions at the intermediate angles and reduced efficiency. The rule presented here of sampling slightly finer than the Nyquist sampling rate has in this case, and others, proven to be reliable in predicting the maximal effective stride.

It is also worth noting that if the right hand side vectors in a block are too far apart, convergence will also slow. It is easy to construct examples where right-hand side vectors belong to orthogonal invariant subspaces of the matrix. In this case, the block algorithm is no more

effective than iterating each right-hand side individually. Therefore, the right-hand side vectors must be close enough together so that their Krylov sub-spaces enrich each other but not so close as to be redundant.

Before the tests shown in this paper were run, it was determined experimentally that requiring that all the relative equation residuals be less than $10^{-2}$ ( $-2$ Bels ) was sufficient for overlay cross sections. With this convergence criterion, the algorithm converged in 10 iterations in all cases. However, this is not a definitive statement on error analysis. The RCS depends only upon the smooth part of the solution, so if the solution error is of high frequency, a "low accuracy" solution can give accurate RCS values. Since the solution error is not normally known (you have to know the exact solution, which we did in this case), one must estimate it from the equation residual.

However, estimating solution error from equation residual is also not straightforward. Understanding it requires explicit knowledge of the eigenvalues and eigenvectors of the MOM matrix. Figures 5, 6 and 7 show in places over 1 Bel (approximately 1 significant digit of accuracy) difference between the relative equation residual and the relative solution error, while in other places the difference is much less. This difference also depends strongly on the completeness of the preconditioner. A convergence criterion which will guarantee accurate solutions or RCS with the minimal number of iterations is still not a reality.

This solution method has been extremely reliable and is becoming the method of choice in our work. We have used it to compute the mono-static RCS for a rank 12,000 MOM matrix on a DEC Alpha workstation. The performance was as presented here, almost an order of magnitude faster than direct LU factorization. Not too long ago, this was a problem that could only be done on a CRAY.

There is significant potential for parallel processing in this algorithm. The block algorithm provides yet another dimension that may be exploited in parallel computation [5] for the algorithm itself and the dense matrix multiplications. More importantly, the block algorithm permits the application of the preconditioner to be parallelized, a significant achievement since the recursive nature of forward reduction and back substitution has hindered attempts at parallelizing this to date.

## References

[1] C. F. Smith, A. F. Petersen, and R. Mittra, "A conjugate gradient algorithm for the treatment of multiple incident electromagnetic fields," *IEEE Trans. Antennas Propagat.*, vol. 37, pp. 1490–1493, November 1989.

[2] V. Simoncini and E. Gallapoulos, "An iterative method for nonsymmetric systems with multiple right-hand sides," *SIAM J. Sci. Comput.*, vol. 16, pp. 917–933, July 1995.

[3] T. F. Chan and W. L. Wan, "Analysis of projection methods for solving linear systems with multiple right-hand sides," *SIAM J. Sci. Comput.*, To Appear.

[4] D. P. O'Leary, "The block conjugate gradient algorithm and related methods," *Linear Algebra Appl.*, vol. 29, pp. 293–322, 1980.

[5] D. P. O'Leary, "Parallel implementation of the block conjugate gradient algorithm," *Parallel Computing*, vol. 5, pp. 127–139, 1987.

[6] W. E. Boyse and A. A. Seidl, "Convergence acceleration by preconditioning a block quasi minimum residual method for a finite element formulation of electromagnetic scattering," in *URSI Symposium Digest*, (Chicago Illinois), p. 172, IEEE-APS/URSI/NEM Joint Symposium, 1992.

[7] W. E. Boyse and A. A. Seidl, "An iterative solver for multiple right hand sides based on the block Lanczos algorithm," in *URSI Symposium Digest*, (Chicago Illinois), p. 173, IEEE-APS/URSI/NEM Joint Symposium, 1992.

[8] W. E. Boyse and A. A. Seidl, "A block QMR method for computing multiple simultaneous solutions to complex symmetric systems," *SIAM J. Sci. Comput.*, January 1996.

[9] Y. Saad and M. H. Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM J. Sci. Stat. Comput.*, vol. 7, pp. 856–869, March 1986.

[10] A. Yeremin, "Recent advances in iterative solution methods: Lectures at NASA/Ames." badger@adonis.ias.msk.su, November 1993.

[11] R. W. Freund, G. H. Golub, and N. M. Nachtigal, "Iterative solution of linear systems," *Acta Numerica*, pp. 57–100, 1991.

[12] Y. Saad, "ILUT: a dual threshold incomplete LU factorization," Research Report UMSI 92/38, University of Minnesota Supercomputer Institute, Minneapolis, MN, March 1992.